Measuring the complexity of the ocean floor

Sams, Thomas; Stage, B.; Agerkvist, Finn T.; Christensen, T.

Published in:
2000 IEEE Workshop on Neural Networks for Signal Processing X

Publication date:
2000

Document Version
Early version, also known as pre-print

MEASURING THE COMPLEXITY OF THE OCEAN FLOOR

T. Sams, B. Stage, F. T. Agerkvist, and T. Christensen
Danish Defence Research Establishment,
Ryvangs Alle 1, 2100 Copenhagen Ø, Denmark
Phone +45 3915 1749
Fax: +45 3929 1533
E-mail: ts@ddre.dk

Abstract. In an attempt to classify the complexity of ocean floor, we suggest a neural-network based algorithm which predicts a pixel based on its surroundings. The mean square of the difference between the original sidescan sonar image and the predicted image is then used as a measure of the complexity of the image.

PROLOGUE

The allocation of routes for ships in potentially mined water is a major challenge in modern warfare. We present an attempt to rank the natural background according to how easy it is to hide mines on the background. In special cases, it may be possible to model the background in great detail, and subtract the model from the original image, thus enhancing objects/structures that do not fit into the background. In the present paper, we present the suggestion to model the background by training a neural network to predict a pixel from its surroundings. The methods discussed are applied to sidescan sonar data.

DESIGN OF NEURAL NETWORK

In order to make a simple predictor for an image based on a feed-forward error back-propagation neural network, we suggest to map the surroundings of the point through a network with one hidden layer to the point. This is illustrated in figure 1. In figure 1b the full architecture, including the hidden layer, is shown.

A fully connected two-layer error-back-propagating network is described here [1]. The input vector v_i is propagated through the network as

$$h_j = \sum_i w_{ji} v_i \tag{1}$$

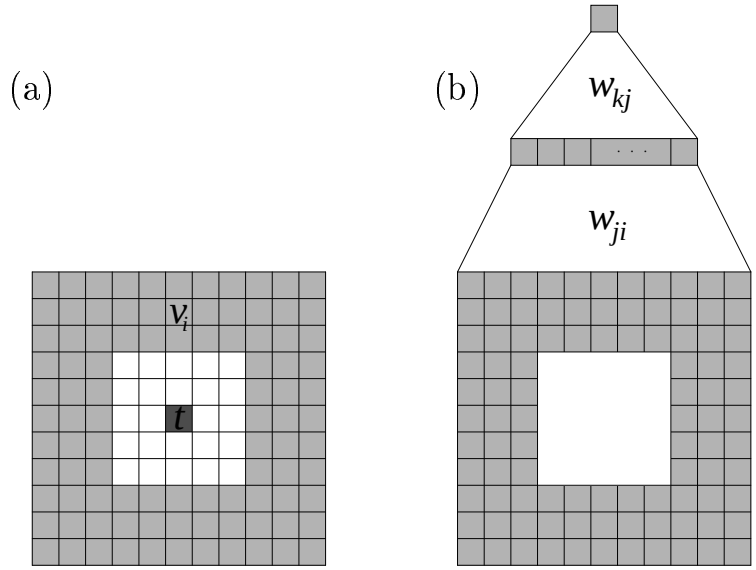

Figure 1: a: Illustration of the trainable filter. The input vector is formed by the image values scaled to the $(-1, 1)$-interval in the light-grey area. The the input vector is propagated through a hidden layer to the target, which is the central pixel. The white area which is ignored should typically have the size of the maximal structure which one wants to separate from the background. In order for the comparison to be fair, the linear filter uses the same input and output regions as are used in the case of the neural network.
b: Complete architecture of the neural network filter. The filter coefficients are adjusted using gradient descent by selecting random input samples from the training set. The performance is evaluated in a separate test region.

$$v_j = g(h_j) \tag{2}$$

$$h_k = \sum_j w_{kj}\, v_j \tag{3}$$

$$v_k = g(h_k) \tag{4}$$

with weights w_{ji} connecting the input to the hidden and w_{kj} connecting the hidden layer to the output v_k.

The activation function is chosen to be the hyperbolic tangent $g(h) = \tanh h$ mapping the signal into the interval $(-1, 1)$. Biases are included by connecting all but the units in the input layer to an extra input fixed at -1. The weights are initialized at normal distributed values with zero mean and variation $1/N_i$ for w_{ji} and variation $1/N_j$ for w_{kj}.

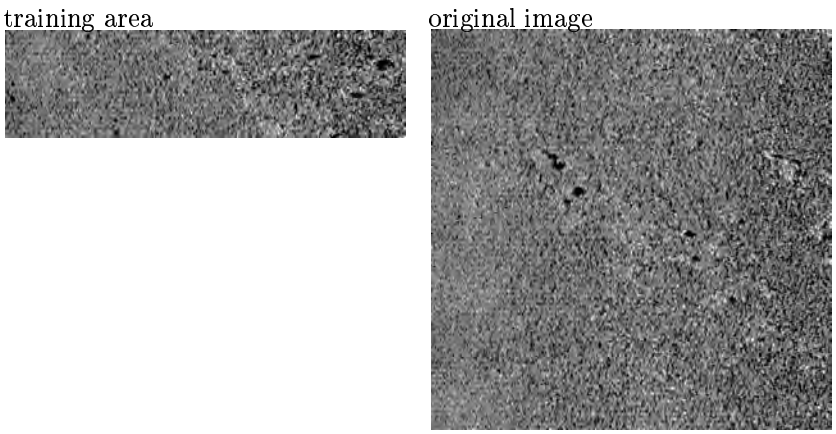

Figure 2: 1) Training area used for training of the neural network and adjustment of the parameters in the linear predictor in the following figures. The global variance in the image is $\sigma^2_{\text{global}} = 2247$.
2) Test region.

During training the weights are changed gradually, following the gradient descent rule, such as to minimize the error function using the gradient descent learning rule. The error function is taken to be the relative entropy

$$S = \sum_v S_v \tag{5}$$

$$S_v = \frac{1}{2} \sum_k \left((1+t_k) \log \frac{1+t_k}{1+v_k} + (1-t_k) \log \frac{1-t_k}{1-v_k} \right) \tag{6}$$

where t_k denotes the desired answer (target) given the input icon v_i producing the output v_k. The error measure diverges when the output is anti-correlated with the target.

In the gradient descent learning rule a weight is changed by an amount proportional to the size of the partial derivative of the error function with respect to the weight. With the entropic error function (5) we find

$$\frac{\partial S}{\partial w_{kj}} = -\delta_k v_j \tag{7}$$

$$\delta_k = t_k - v_k \tag{8}$$

$$\frac{\partial S}{\partial w_{ji}} = -\delta_j v_i \tag{9}$$

$$\delta_j = (1 - v_j^2) \sum_k w_{kj} \delta_k \tag{10}$$

describing the back propagation of the error. This generalizes to more layers by duplicating equations (9) and (10) for each extra layer.

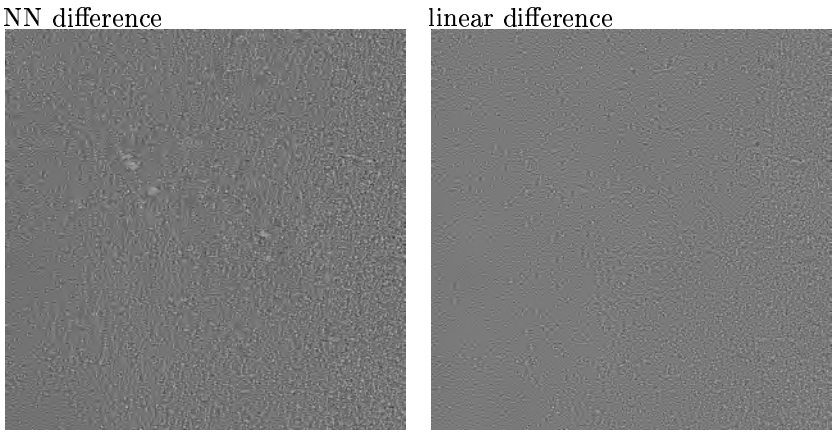

Figure 3: Comparison between the performance of a NN filter of size 7×7 with a 1×1 opening and a linear filter with the same geometry. The neural network was trained with 2 units in the hidden layer. Both the neural network and the linear predictors had their parameters adjusted in the training region. The mean-square of the difference images are $\sigma^2_{\mathrm{NN}} = 650$ and $\sigma^2_{\mathrm{linear}} = 655$ respectively, i. e. the predictions are of the same quality. The contrast in the difference images has been scaled a factor 2 relative to the original image. The mean-square of the difference between the average in the input region to the filter and the central pixel $\sigma^2_{\mathrm{aver}} = 1849$.

In order to damp oscillations during training we added a momentum term so the weight change gets some memory of previous time steps[2]. The weight is then changed

$$\Delta w(t) = -\eta \frac{\partial S}{\partial w}(t) + \alpha \, \Delta w(t-1) \tag{11}$$

at update t. Training is typically convergent with $\eta = 0.0001$. The momentum parameter is set to $\alpha = 0.9$.

LINEAR PREDICTION OF PIXELS

In order to evaluate the performance of the non-linear predictor, we compare to the performance of a linear predictor. The review of the linear predictor below below serves to establish the formalism. The filter may be viewed as that shown in figure 1. The position of the filter is labelled p and the pixels in the input region are labelled i or j. If we let α be the linear expansion coefficients, the predicted target (central) pixel may be written

$$\tilde{t}_p = \sum_i \alpha_i v_{ip}. \tag{12}$$

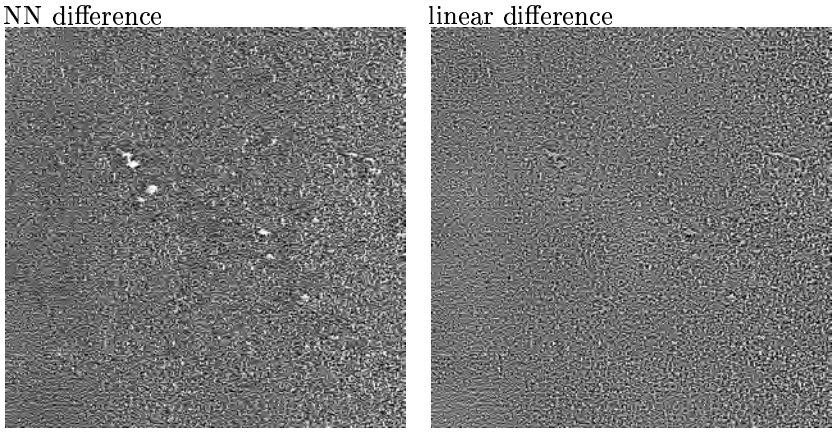

Figure 4: Comparison between the performance of a NN filter of size 9×9 with a 3×3 opening and a linear filter with the same geometry. The neural network was trained with 2 units in the hidden layer. Both the neural network and the linear predictors had their parameters adjusted in the training region. The mean-square of the difference images are $\sigma^2_{\mathrm{NN}} = 1576$ and $\sigma^2_{\mathrm{linear}} = 1616$ respectively, i.e. the predictions are of the same quality. The contrast in the difference images has been scaled a factor 2 relative to the original image. The mean-square of the difference between the average in the input region to the filter and the central pixel $\sigma^2_{\mathrm{aver}} = 2116$.

We define an error function

$$E = \sum_p (t_p - \tilde{t}_p)^2 \tag{13}$$

which is minimal when

$$\frac{\partial E}{\partial \alpha_i} = 2 \sum_p \left(\sum_j \alpha_j v_{jp} - t_p \right) v_{ip} \tag{14}$$

$$= 2 \sum_j \alpha_j \sum_p v_{jp} v_{ip} - 2 \sum_p v_{ip} t_p \tag{15}$$

vanishes for all i. Defining the signal correlation matrix

$$\sigma_{ji} = \sum_p v_{jp} v_{ip} \tag{16}$$

and the target-signal correlation vector

$$r_i = \sum_p v_{ip} t_p \tag{17}$$

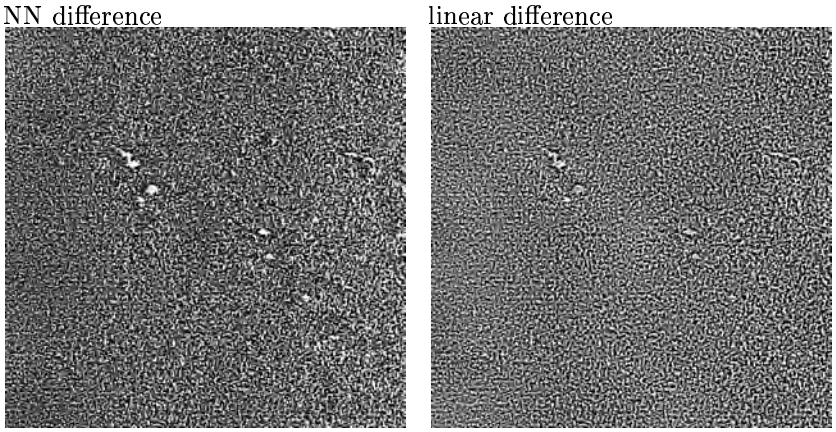

Figure 5: Comparison between the performance of a NN filter of size 9×9 with a 5×5 opening and a linear filter with the same geometry. The neural network was trained with 2 units in the hidden layer. Both the neural network and the linear predictors had their parameters adjusted in the training region. The mean-square of the difference images are $\sigma^2_{\rm NN} = 2070$ and $\sigma^2_{\rm linear} = 2098$ respectively, i.e. the predictions are of the same quality. The contrast in the difference images has been scaled a factor 2 relative to the original image. The mean-square of the difference between the average in the input region to the filter and the central pixel $\sigma^2_{\rm aver} = 2227$.

we find

$$\sum_j \alpha_j \sigma_{ji} = r_i \tag{18}$$

$$\alpha_i = \sum_j r_j \sigma^{-1}_{ji}. \tag{19}$$

The difference between the signal and the linear prediction is then

$$\Delta t_p = t_p - \tilde{t}_p \tag{20}$$

$$= t_p - \sum_{ij} r_j \sigma^{-1}_{ji} v_{ip} \tag{21}$$

$$= t_p - \sum_{ijq} t_q v_{jq} \sigma^{-1}_{ji} v_{ip}. \tag{22}$$

Note that the number of positions P used to determine the filter must be larger than I.

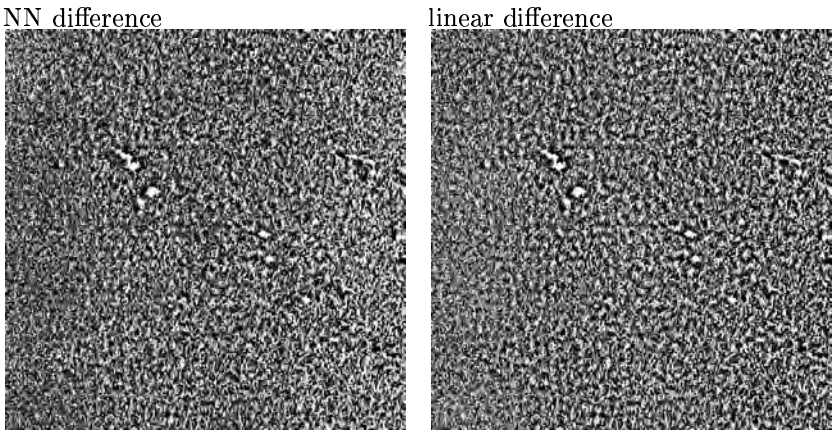

Figure 6: Comparison between the performance of a NN filter of size 19×19 with a 15×15 opening and a linear filter with the same geometry. The neural network was trained with 2 units in the hidden layer. Both the neural network and the linear predictors had their parameters adjusted in the training region. The mean-square of the difference images are $\sigma^2_{\mathrm{NN}} = 2209$ and $\sigma^2_{\mathrm{linear}} = 2209$ respectively, i. e. the predictions are of the same quality. The contrast in the difference images has been scaled a factor 2 relative to the original image. The mean-square of the difference between the average in the input region to the filter and the central pixel $\sigma^2_{\mathrm{aver}} = 2209$.

THE MEASURE OF COMPLEXITY

After having trained the neural network in a training region, it is used to make predictions in a similar, but separate test region. The mean-square deviation between the prediction and the original images is taken as a measure of complexity. The hope was that the non-linear predictor would perform significantly better than a linear predictor.

PERFORMANCE OF THE NN PREDICTOR COMPARED TO THE LINEAR PREDICTOR

In figure 2 we show the training and test regions used in the comparison of the performance of the neural network and the linear predictors. In figures 3-6 is shown the resulting differences between prediction and original for different openings of the filters. In all cases, the neural network and the linear predictors give rise to practically identical measures of complexity.

We have performed the same analysis on various examples of seabed: sand ripples, trawl tracks, stones, and seaweed. In all cases the linear predictor performs equally well as the neural network predictor.

DISCUSSION AND FUTURE WORK

The linear as well as the neural network auto predictors perform poorly when the input filter is opened. One possible explanation could be that, when trying to make a linear or nonlinear fit from points far from the central point, good knowledge of higher order derivatives of the field become important. This, on the other hand, puts high demands on the quality of the data. The sidescan sonar data, we have had access to have been of low quality, with only 4 bits of information in each pixel, saturation problems, and systematic errors. It is of importance for the evaluation of the NN and linear predictors to test the methods with data of better quality.

We are still in the process of clarifying how the measures of complexity on different scales should be normalized and possibly be compared. We expect to be able to present these results at the NNSP 2000 conference. Further, we are in the process of getting better data for the evaluation of the measures of complexity presented here and expect to present a better analysis using data with less noise than we have shown here.

EPILOGUE

We have investigated the possibility to use a neural network as a tool to predict a central region from the surroundings to create a good measure of complexity of an image. We conclude that the suggested neural network performs no better than a linear predictor when used on a sonar image. Thus, the linear predictor should be preferred.

REFERENCES

[1] J. A. Hertz, A. Krogh and R. G. Palmer, **Introduction to the Theory of Neural Computation**, 350 Bridge Parkway, Redwood City, CA 94065: Addison-Wesley, 1990.

[2] D. Plaut, S. Nowlan and G. Hinton, "*Experiments on learning by back propagation*," Technical Report CMU-CS-86-126, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1986.